Adaptive Software for a Changing World

Analyzing Properties of Service Discovery Protocols Using an Architecture-Based Approach

Christopher Dabrowski and Kevin Mills

National Institute of Standards and Technology

Gaithersburg, MD USA

Presentation at

Working Conference on

Complex and Dynamic Systems Architectures

Brisbane, Australia

December 12, 2001

	Report Documentation Page		*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE **12 DEC 2001**	2. REPORT TYPE	3. DATES COVERED **00-00-2001 to 00-00-2001**

4. TITLE AND SUBTITLE **Analyzing Properties of Service Discovery Protocols Using an Architecture-Based Approach**	5a. CONTRACT NUMBER
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **National Institute of Standards and Technology,Gaithersburg,MD,20899**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
Presentation at Working Conference on Complex and Dynamic Systems Architectures, Brisbane, Australia, December 12, 2001

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**27**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

Presentation Roadmap

- Project Objectives, Motivation, and Goals

- Modeling & Analysis
  - Architecture-based approach
  - Generic UML structural model
  - Specific models instantiated with Architecture Description Language

- Assessment of Architecture-Based Approach

- Overview of On-Going Work
  - Measuring responses of different service discovery architectures to node and link failures
  - How can these responses be improved?

- Plans for Future Work

Dynamic discovery protocols..

enable **network elements** (including software clients and services, and devices):
  (1) to **discover** each other without prior arrangement,
  (2) to **express** opportunities for collaboration,
  (3) to **compose** themselves into larger collections that cooperate to meet an application need, and
  (4) to **detect and adapt to changes** in network topology.

Selected Current (First) Generation Protocols for Dynamic Service Discovery
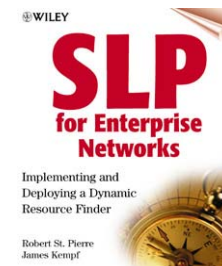
Our Goal

1) Use ADLs and associated tools to **analyze Discovery Protocol specifications** to assess consistency and completeness wrt dynamic change conditions—<u>basis for defining gauges</u>.

2) To provide metrics and approaches to **compare and contrast emerging** commercial service **discovery technologies** with regard to critical functions, structure, behavior, performance and scalability in the face of dynamic change and to strengthen the robustness, quality and correctness of designs for future protocols.

3) Provide recommendations on improving ADLs as tools for analyzing architectures under conditions of dynamic change.

Our Overall Technical Approach

- Build a **generic, domain model** (UML) providing consistent terminology encompassing a range of service discovery protocols.
- Build **executable models** of service discovery protocols from extant specifications--for analysis under conditions of dynamic change.
- Define **consistency conditions** and **metrics** to assess the performance of executable models.
- Use **scenarios** to exercise models conditions of dynamic change.
- **Compare and contrast** our **models** with regard to function, structure, behavior, performance, complexity, and scalability under conditions of dynamic change.
- Design, model, and evaluate **protocol mechanisms** that enable discovery protocols to self-adapt in the face of dynamic change *(this part of the project is funded by the DARPA Fault Tolerant Networks program).*

Summary of Current Results

- Developed ***architecture-based approach*** for modeling service discovery protocols that relies on
  - **property analysis** using consistency conditions to assess robustness of distributed software components to dynamic change
  - ***event analysis*** using explanatory capabilities provided by ADLs (*Rapide*)

  to analyze ***consistency and completeness*** of software specifications under conditions of dynamic change.
- Demonstrated viability of the approach to ***analysis of behavior and performance*** of commercial Service Discovery Protocol specification.
- Evaluated ADLs for use in modeling and analyzing dynamic distributed systems and provided recommendations
- Extended approach to make quantitative measurements of response of alternative service discovery architectures to dynamic change (currently being applied in ongoing study).

Presentation Roadmap

- Project Objectives, Motivation, and Goals

- Modeling & Analysis
  - Architecture-based approach
  - Generic UML structural model
  - Specific models instantiated with Architecture Description Language

- Assessment of Architecture-Based Approach

- Overview of On-Going Work
  - Measuring responses of different service discovery architectures to node and link failures
  - How can these responses be improved?

- Plans for Future Work

Sample Network Topology Applicable to Jini Entities

Discovers SCMs and registers services

Aggressive Discovery   Multicast Group

Service Manager (SM)

Remote Method Invocation

Service Cache Manager (SCM)

Unicast Links

Service User (SU)

Lazy Discovery   Multicast Group

Matches services registered by SM to notification requests registered by SU

Discovers SCMs and registers notification requests.
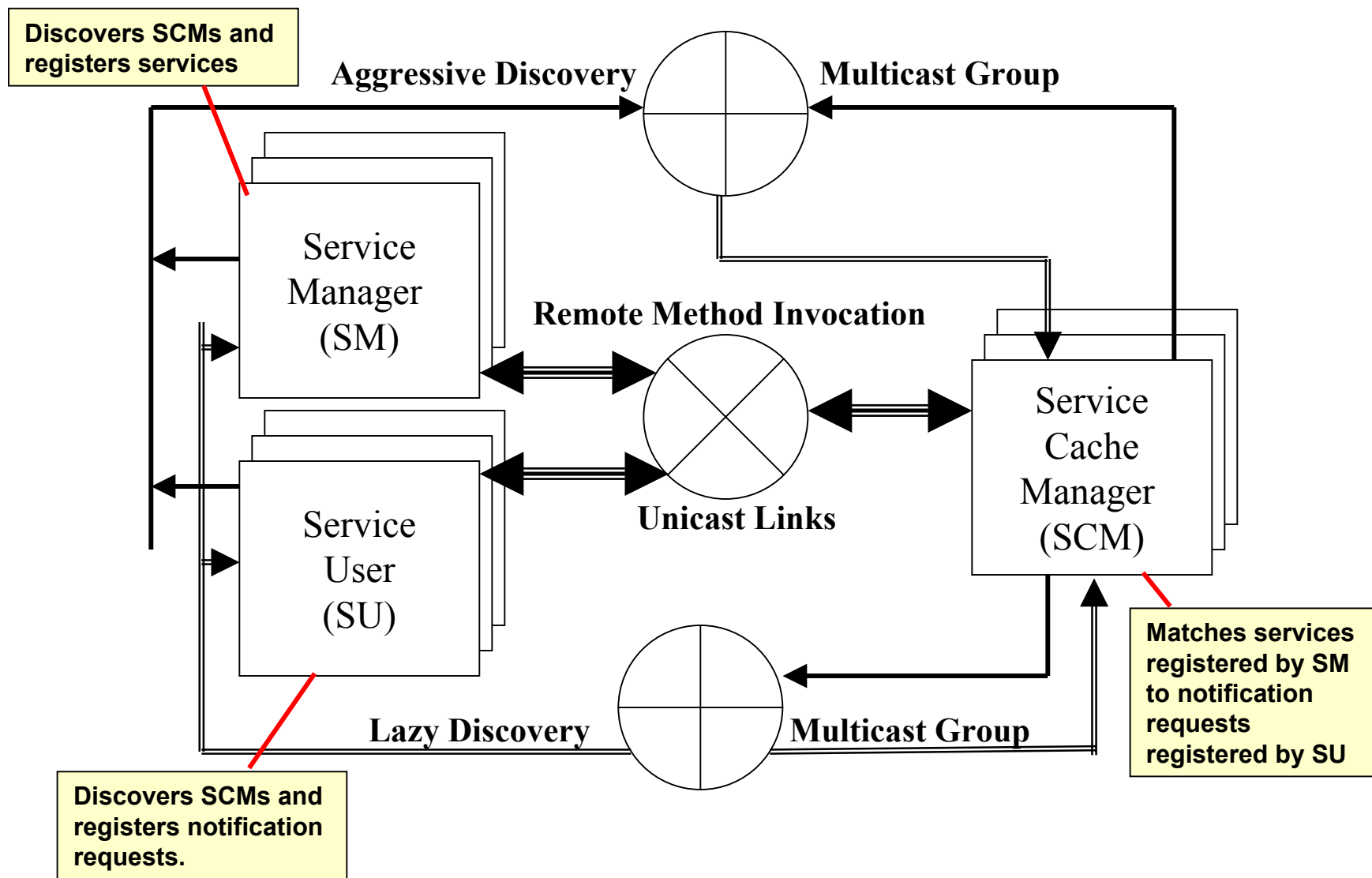
Foundation : A Generic Structural Model (UML) for Service-Discovery Domain
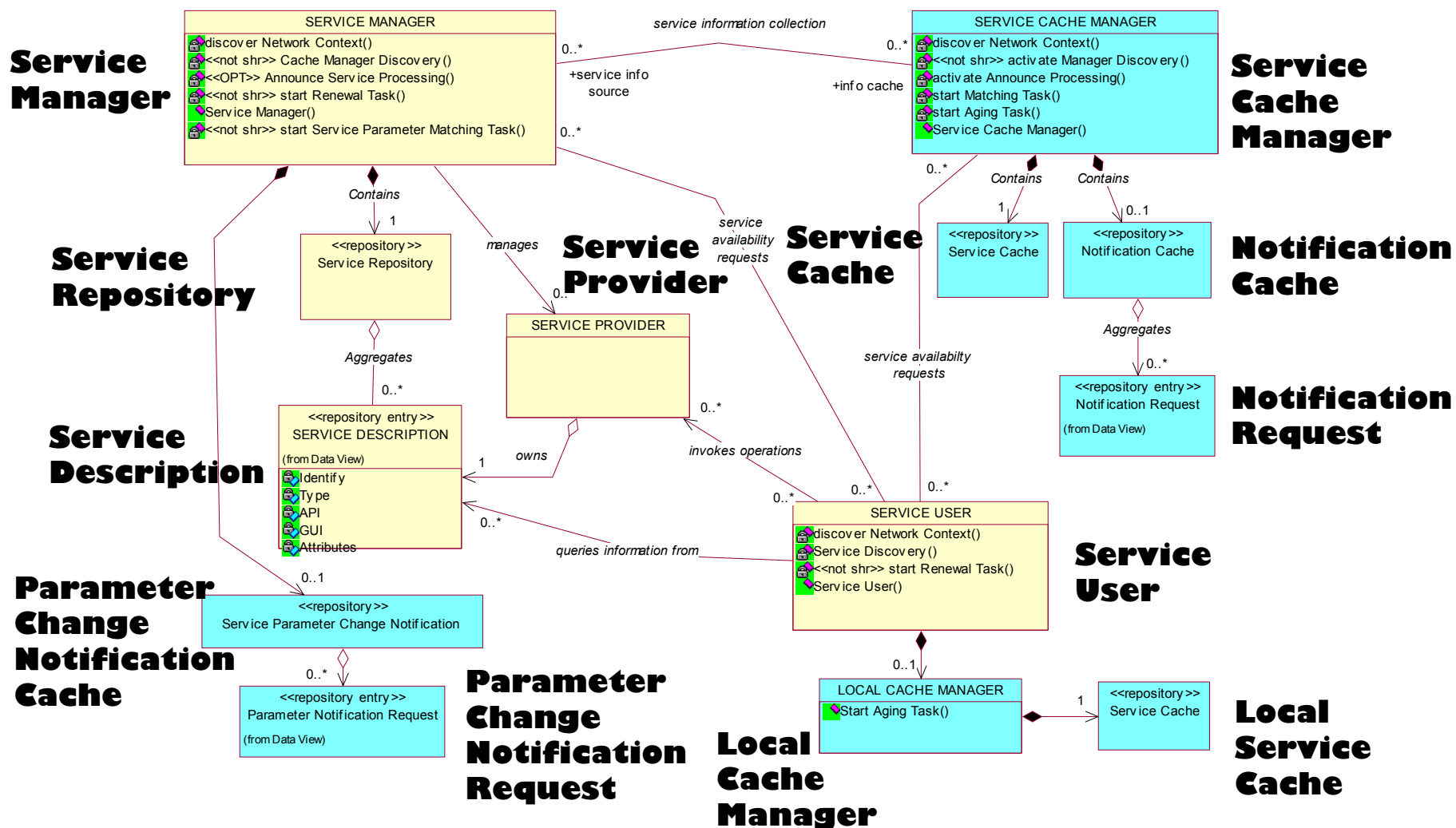
Architectural Description Languages & Tools….

- **Represent essential complexity** of service discovery protocols with effective abstractions
  - *Rapide*, public-domain ADL and toolset developed at Stanford University for DARPA, provides ability to execute architecture specifications, producing Partially Ordered Sets of Events (POSETs) for analysis.
- Provide a framework and context
  - to **define metrics** that yield qualitative and quantitative measures of dynamic component-based software
  - to **model alternate approaches** to specific functions or mechanisms
  - to help **pinpoint** where **inconsistencies and ambiguities** may exist within software implementing specifications & to understand how such issues arise
  - to **compare and contrast** dynamic **service discovery architectures**

Architecture-based Approach to Modeling and Analysis

(using Rapide, an Architecture Description Language and Tools Developed for DARPA by Stanford)

Time	Command	Parameters
5	NodeFail	SM4
5	LinkFail	SCM1 SM4
10	GroupJoin	SM4 GROUP1
10	FindService	SU8 5 1 2 S XYZ ALL
50	AddService	SM4 SCM3 T ATT API GUI 20 30

Scenario

Topology

Aggressive Discovery   Multicast Group

Service Manager

Remote Method Invocation

Service User

Service Cache Manager

Unicast Links

Lazy Discovery   Multicast Group

Specification Model

Execute with Rapide

```
-- ****************************************************
-- ** 3.3  DIRECTED DISCOVERY CLIENT INTERFACE     **
-- ****************************************************
-- This is used by all JINI entities in directed
-- discovery mode.  It is part of the SCM_Discovery
-- Module. Sends Unicast messages to  SCMs on list of
-- SCMS to be discovered until all SCMS are found.
-- Receives updates from SCM DB of discovered SCMs and
-- removes SCMs accordingly
-- NOTE: Failure and recovery behavior are not
-- yet defined and need reviw.
TYPE Directed_Discovery_Client
  (SourceID : IP_Address; InSCMsToDiscover : SCMList; StartOption : DD_Code;
   InRequestInterval : TimeUnit; InMaxNumTries : integer; InPV : ProtocolVersion)
IS INTERFACE
SERVICE DDC_SEND_DIR   : DIRECTED_2_STEP_PROTOCOL;
SERVICE DISC_MODES     : dual SCM_DISCOVERY_MODES;
SERVICE DD_SCM_Update  : DD_SCM_Update;
SERVICE SCM_Update     : SCM_Update;
SERVICE DB_Update      : dual DB_Update;
SERVICE NODE_FAILURES : NODE_FAILURES;  -- events for failure and recovery.
ACTION
 IN Send_Requests(),
   BeginDirectedDiscovery();
BEHAVIOR
  action animation_Iam (name: string);
  MySourceID    : VAR IP_Address;
  PV            : VAR ProtocolVersion;
```
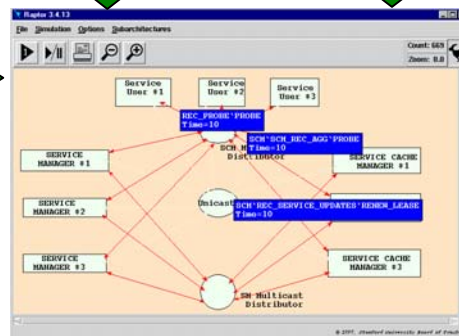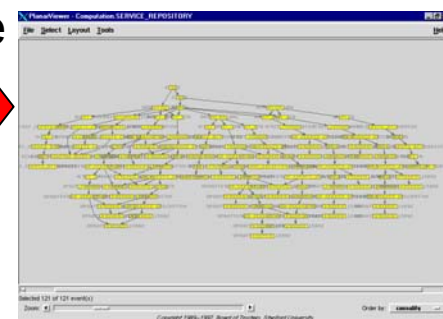
Consistency Conditions

Analyze POSETs

For All (SM, SD, SCM):
(SM, SD) IsElementOf SCM registered-services   (CC1)
implies SCM IsElementOf SM discovered-SCMs

For All (SM, SD, SCM):
SCM IsElementOf SM discovered-SCMs &   (CC2)
(SD) IsElementOf SM managed-services
implies (SM, SD) IsElementOf SCM registered-services

For All (SM, SD, SCM):
SCM IsElementOf SM discovered-SCMs &   (CC3)
(SM, SD) IsElementOf SCM registered-services &
NOT (SCM IsElementOf SM persistent-list)
implies Intersection (SM GroupsToJoin, SCM GroupsMemberOf)

For All (SM, SD, SCM, SU, NR):
(SU, NR) IsElementOf SCM requested-notifications &   (CC4)
(SM, SD) IsElementOf SCM registered-services &
Matches((SM, SD), (SU,NR))
implies (SM, SD) IsElementOf SU matched-services
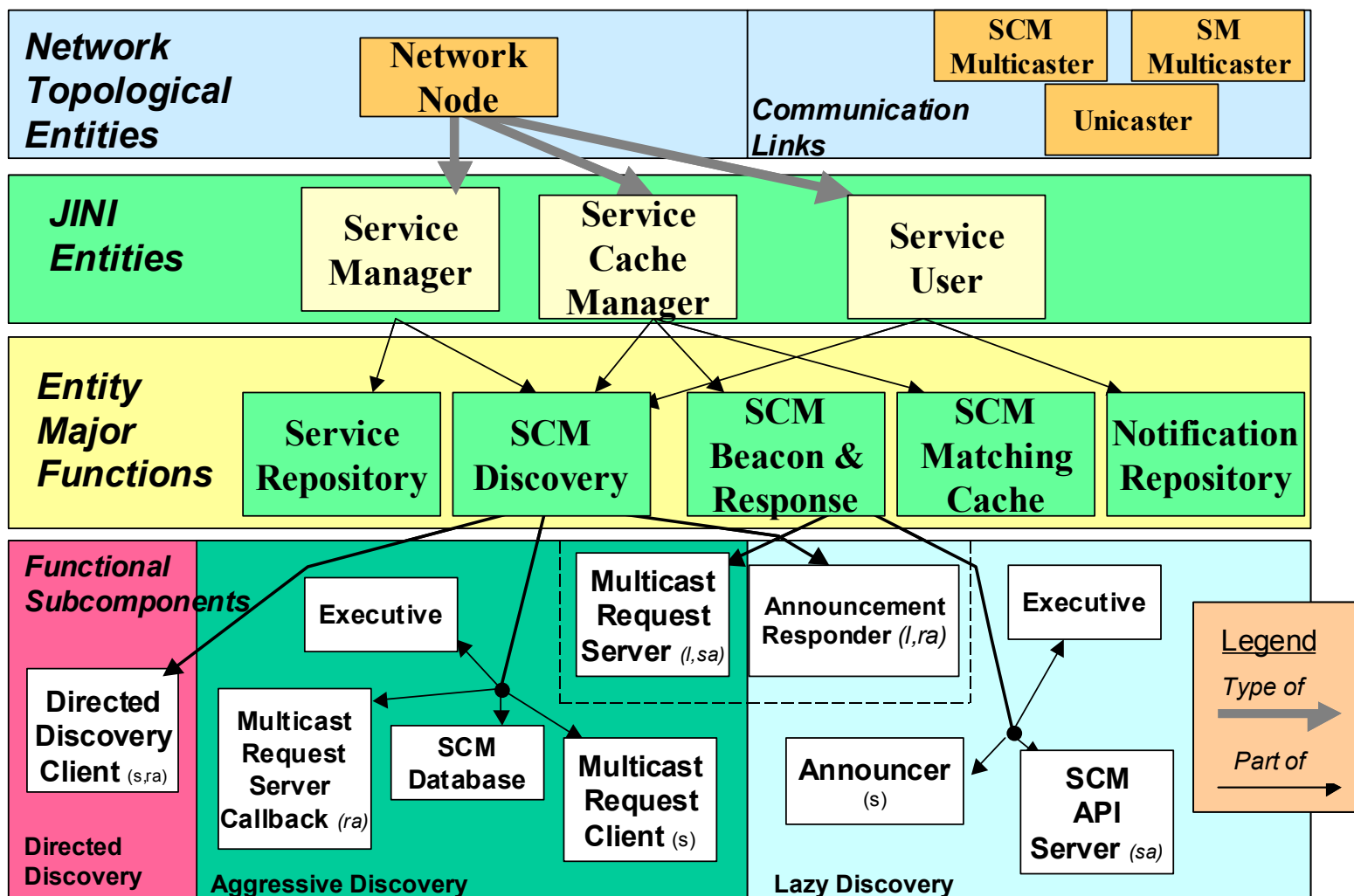
Assess Correctness, Performance, & Complexity

Layered View of Prototype JINI Architecture in Rapide
Derived from SEI Architectural Layers Approach

NIST
National Institute of Standards and Technology
Technology Administration, U.S. Department of Commerce

NIST CENTENNIAL
1901-2001

DARPA

Real-Time Checking of Consistency Conditions

Sample Consistency Condition (CC #4 race condition)

For All **(SM, SD, SCM, SU, NR):**
    (SU, NR) *IsElementOf* **SCM requested-notifications &**
    (SM, SD) *IsElementOf* **SCM registered-services &**
    Matches ((SM, SD), (SU, NR))
        implies **(SM, SD)** *IsElementOf* **(SU matched-services)**

…that is, if an SU has requested notification with a Service Cache Manager of a service that matches a service description registered by a Service Manager on the same Cache Manager, then that service description should be provided to the Service User.

*Assuming absence of network failure and normal delays due to updates

- **SM is Service Manager**
- **SD is Service Description**
- **SCM is Service Cache Manager**
- **SU is Service User**
- **NR is Notification Request**

- **requested-notifications is a set of (SU,NR) pairs maintained by the SCM**
- **registered-services is a set of (SM,SD) pairs maintained by the SCM**
- **matched-services is the set of (SM,SD) pairs maintained by the SU**

Use of Property and Event Analysis to Identify and Understand Possible Registration Race Condition

For All (SM, SD, SCM, SU, NR):

    (SU, NR) IsElementOf SCM requested-notifications &   (CC4)

    (SM, SD) IsElementOf SCM registered-services &

    Matches((SM, SD), (SU,NR))

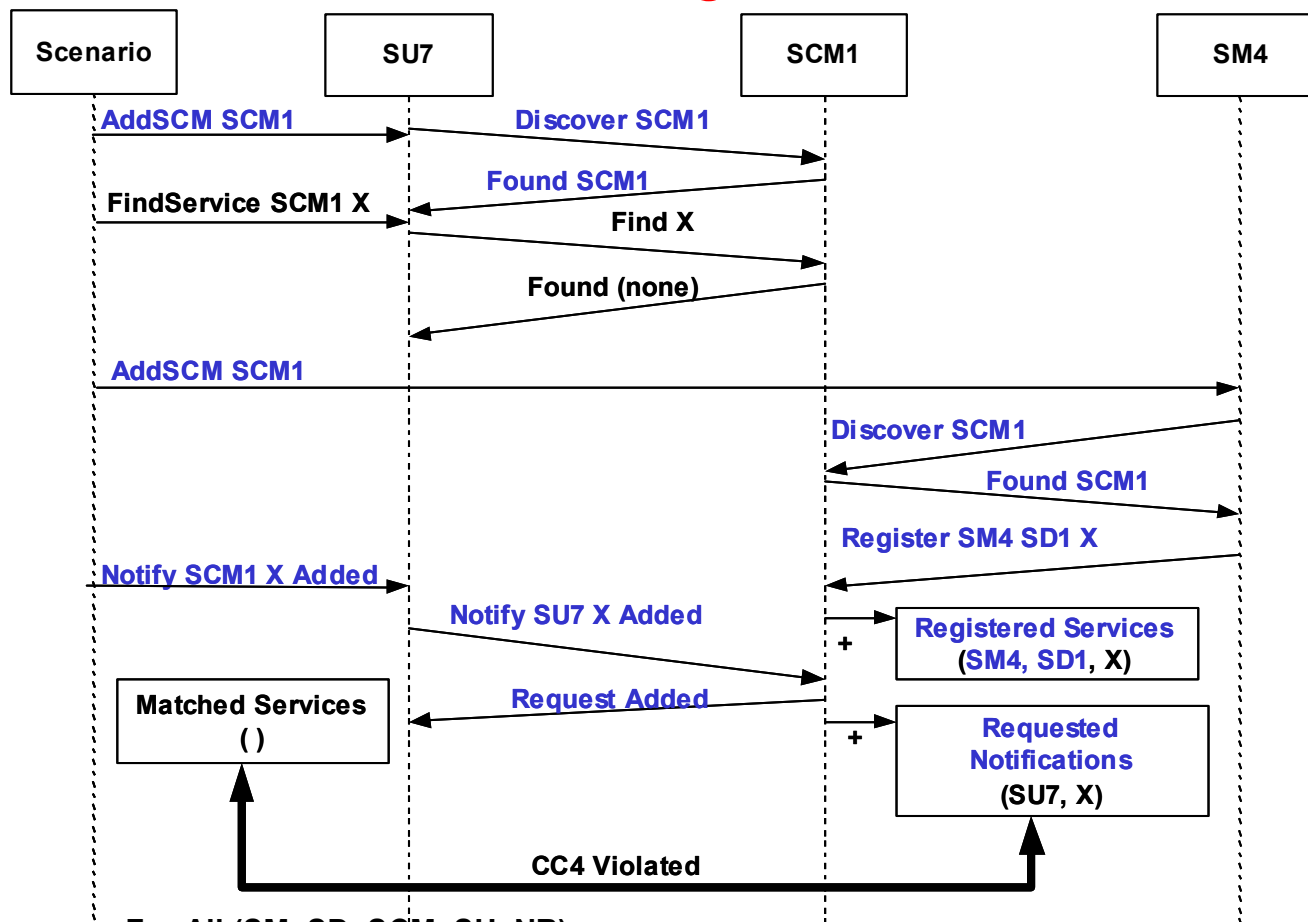   implies (SM, SD) IsElementOf SU matched-services

Presentation Roadmap

- Project Objectives, Motivation, and Goals

- Modeling & Analysis
  - Architecture-based approach
  - Generic UML structural model
  - Specific models instantiated with Architecture Description Language

- Assessment of Architecture-Based Approach

- Overview of On-Going Work
  - Measuring responses of different service discovery architectures to node and link failures
  - How can these responses be improved?

- Plans for Future Work

Assessment of Architecture-Based Approach

- Used approach to verify robustness of Jini Protocol under variety of failure scenarios

- Merits of approach for analysis of dynamic behavior
  - Architectural model allowed easier representation of discovery components and their behavior: more precise, concise and informative.
  - Provided insight into, and understanding of, collective behavior of interacting components than could static specification
  - Able to identify areas of ambiguity, inconsistency, and incompleteness (reported 4 instances)
  - Single model can be analyzed for behavior, performance and logical properties
  - Allowed alternative implementation options to be considered and explored using realistic scenarios

Assessment of Architecture-Based Approach (con't)

- Areas for Improvement
  - Improvements to representation of structure
    - Benefits of using first-class connectors
    - Relaxation of strictly hierarchical connectivity
      - Greater fidelity to real-world designs
      - Fewer events in POSET
  - Improvements to representation of behavior
    - Explicit definition of component state (through export of selected state variables) -- on par with definition of events
    - Evaluation of consistency conditions against state(s) across multiple components
    - Linkage of events to state
  - Need for customizable domain-specific syntax
    - Improve understandability of ADL specifications to non-specialists w/ customizable domain-specific syntax.

Presentation Roadmap

- Project Objectives, Motivation, and Goals

- Modeling & Analysis
  - Architecture-based approach
  - Generic UML structural model
  - Specific models instantiated with Architecture Description Language

- Assessment of Architecture-Based Approach

- Overview of On-Going Work
  - Developing metrics to measure responses of different service discovery architectures to node and communication failures
  - How can these responses be improved?

- Plans for Future Work

Focus: How do Two- and Three-Party Architectures for Service Discovery Respond to Failures?

- **Two-Party vs. Three-Party architectures**
  Two alternative architectural designs that underlie commercial service discovery protocols, including Jini, UPnP, and Service Location Protocol

- **Impact of Study:**
  1. Develop and formalize **metrics and related generic set of test scenarios** that can be used to develop and test service discovery products/applications
  2. Continue to provide **recommendations on improving ADLs**
  3. Provide valuable information to designers and users of service discovery protocols for **improving specifications**, thus **promoting software quality and reliability**.

Two Party vs. Three Party Architectures

Third Party

Service Manager

SERVICE MANAGER
- discover Network Context()
- <<not shr>> Cache Manager Discovery()
- <<OPT>> Announce Service Processing()
- <<not shr>> start Renewal Task()
- Service Manager()
- <not shr> start Service Parameter Matching Task()

service information collection

0..* +service info source

0..*

Service Cache Manager

SERVICE CACHE MANAGER
- discover Network Context()
- <<not shr>> activate Manager Discovery()
- activate Announce Processing()
- start Matching Task()
- start Aging Task()
- Service Cache Manager()

0..* +info cache

Service Cache

0..*

Contains   Contains

1   0..1

<<repository>> Service Cache

<<repository>> Notification Cache

Notification Cache

Contains   1

<<repository>> Service Repository

manages

Service Repository

Service Provider

SERVICE PROVIDER

service availability requests

Aggregates

0..*

Aggregates   0..*

<<repository entry>> Notification Request (from Data View)

Notification Request

0..*

Service Description

<<repository entry>> SERVICE DESCRIPTION (from Data View)
- Identify
- Type
- API
- GUI
- Attributes

1   owns

0..*

service availabilty requests

0..*   0..*   0..*

0..*

invokes operations

0..1

<<repository>> Service Parameter Change Notification

queries information from

Service User

SERVICE USER
- discover Network Context()
- Service Discovery()
- <<not shr>> start Renewal Task()
- Service User()

Parameter Change Notification Cache

0..*

<<repository entry>> Parameter Notification Request (from Data View)

Parameter Change Notification Request

0..1

LOCAL CACHE MANAGER
- Start Aging Task()

Local Cache Manager

1

<<repository>> Service Cache

Local Service Cache

Selected Current (First) Generation Protocols for Dynamic Service Discovery

JINI

3-Party Design

Universal

Plug and Play

2-Party Design

SLP for Enterprise Networks
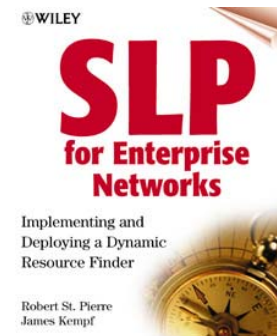
Adaptive 2/3-Party Design

The Salutation Consortium

Vertically Integrated 3-Party Design

HAVi

Network-Dependent 3-Party Design

Bluetooth

Network-Dependent 2-Party Design

How Do Service Discovery Architectures Propagate Changes During Communication Failures?

- **Change Propagation and Consistency Maintenance:** In both two-party and three-party architectures changes in critical characteristics of Service Descriptions (SDs) must propagate from Service Managers (SMs) to Service Users (SUs) that already hold copies of the SDs.
  - Change propagation may take place through polling, eventing, or ad-hoc announcements – How do these strategies compare?
  - Does the existence of a third party (i.e., Service Cache Manager, or SCM) improve or hinder performance?
- **Approach to metrics:** use **property analysis** and **failure test scenarios** to compare and contrast the alternative architectures wrt
  - **Probability of residual inconsistency** – probability that $SM_k(SD_i)$ *not equal to* $SU_j(SD_i)$ for a specific i, j, k within a target time bound.
  - **Change propagation latency** - time delay from [$SM_k(SD_i)$ *not equal to* $SU_j(SD_i)$] until [$SM_k(SD_i)$ *equal to* $SU_j(SD_i)$]
  - **Change propagation overhead** - number of messages in interval from [$SM_k(SD_i)$ *not equal to* $SU_j(SD_i)$] until [$SM_k(SD_i)$ *equal to* $SU_j(SD_i)$]
- Use **event analysis** to understand why different architectures and consistency maintenance strategies vary

How Do Service Discovery Architectures Recover Consistency After Communication and Node Failures?

- **Discovery and Recovery**: In both two-party and three-party architectures, SMs, SUs, and SCMs (where applicable) strive to maintain consistent descriptions (SDs) about discovered services and about event notifications. Link and node failures may lead to temporary loss of information about discovered services. Once failures are repaired, the information must be recovered.

- **We seek to develop metrics** to compare and contrast different service-discovery architectures and specifications. For example:

  - How do discovery latencies and overheads compare?

  - How do event registration latencies and overheads compare?

 - How do recovery latencies and overheads compare?

Presentation Roadmap

- Project Objectives, Motivation, and Goals

- Modeling & Analysis
  - Architecture-based approach
  - Generic UML structural model
  - Specific models instantiated with Architecture Description Language

- Assessment of Architecture-Based Approach

- Overview of On-Going Work
  - Measuring responses of different service discovery architectures to node and link failures
  - How can these responses be improved?

- Plans for Future Work

Extending UML Model to Encompass Message Exchanges and Assertions
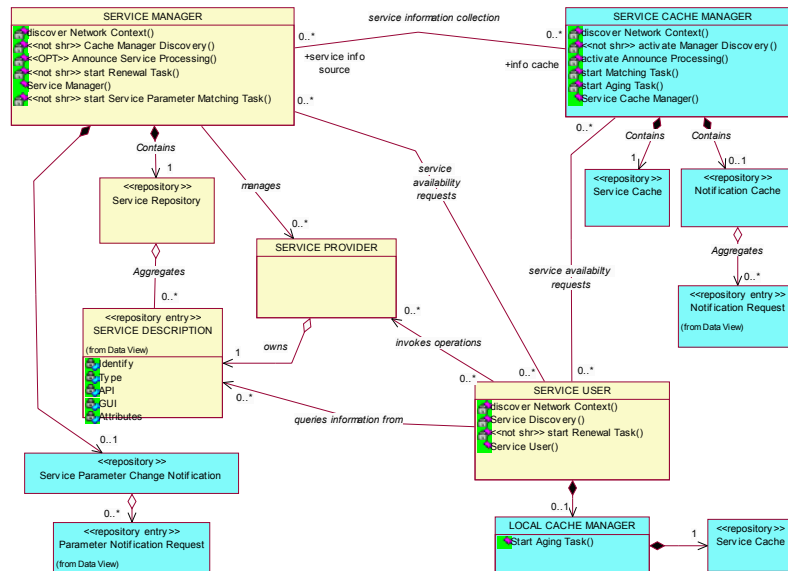
+ Message Set
+ Consistency Conditions
   and other assertions

that capture commonality and variability in Service Discovery Domain

--> Reformulate Rapide-based models in terms of this generic model of structure and behavior
--> Using this model as a basis, develop metrics for more precise assessment of Service Discovery Architectures and *assist in development of future specifications and new designs.*

Investigating Metrics and Use of Architectural Models to Measure System Complexity

- Using the unified architectural model, augment basic set of metrics by creating representations of complexity metrics proposed in the literature, such as
  - algorithmic information complexity [Gammerman and Vovk] [Kolmogorov], [Solomonoff]
  - cyclomatic complexity [McCabe]
  - others (to be selected)

To Delve More Deeply

Currently Available Paper

- Christopher Dabrowski and Kevin Mills, "Analyzing Properties and Behavior of Service Discovery Protocols using an Architecture-based Approach", accepted at DARPA-sponsored *Working Conference on Complex and Dynamic Systems Architecture*.

Available Software Artifacts

- Generic UML Structural Model (in Rational Rose format) of Discovery Protocols, including specific projections to Jini, UPnP, and SLP
- Rapide Models of Jini and UPnP (*in progress*).
- SLX Simulation Model of UPnP (*in progress*).

Related Web Sites

- http://www.itl.nist.gov/div897/ctg/adl/sdp_projectpage.html
- http://w3.antd.nist.gov/net_pc.shtml